Some Things You Always Wanted to Know About Web Pages

(But Were Too Busy to Ask)

Simon Schubert
EPFL, Lausanne
simon.schubert@epfl.ch

Willy Zwaenepoel
EPFL, Lausanne
willy.zwaenepoel@epfl.ch

ABSTRACT

The organic growth of the web has led to web sites that exhibit a large variety of properties. We conduct a large-scale study to gain quantitative insights into the browser-side effects of the structure and behavior of thousands of the most popular web sites. We find that 50 % of web pages load more than 100 resources from more than 17 distinct hosts and make the browser process more than 730 kB of javascript.

Embedded third-party advertisements are the prevailing model on the web to monetize services or content that are otherwise provided free-of-cost to end users. We quantify the effects on browser-side resource consumption caused by advertisements and find that on pages with ads, the ads make up 20 % of all HTTP requests. Furthermore, ads lead to three times as many javascript background events that prevent the processor from staying in deep sleep mode.

Keywords

statistics, web pages, browser side, advertisements

1. INTRODUCTION

During the last 10 years, the complexity of web pages has increased dramatically. Static text-centric pages and "dynamic" PHP sites have given way to media-heavy pages that are not only loaded with images and videos, but are also riddled with advertisements, social media integration and invisible tracking of browsing behavior.

A lot of time spent with computers these days is happening on the web, mediated by browsers. In a sense, the browser can be seen as the modern day operating system, with web pages being the counterpart to traditional programs. However, this analogy breaks down when comparing usage patterns: the traditional use of a single program at a time is contrasted by multi-tabbed browsing and short web page visits. This makes it essentially impossible for a user to understand the effects of web pages on their computer system.

Other than the comparatively small set of traditional applications, the number of web sites is effectively quickly approaching practical infinity, all with a different composition of media, style sheets, javascript and advertisements. We set out to characterize a large set of web sites to understand common trends among them, the extent of their differences, and the role of advertisements on important browser side metrics.

2. METHODOLOGY

We now describe how we collect the base data set that is used to extract the analysis in this paper. On a high level, we automatically direct instances of an actual browser to a large set of web page addresses. Each browser instance is created fresh, with empty cache and cookie storage. The instance stays a fixed time on the web page before being shut down. We capture trace information from the browser during its entire life time.

Selection of Addresses

This survey of web pages is not intended to produce results proportional to "typical" browsing behavior; instead, we are interested in providing an analysis of web pages per se. Because there is an essentially infinite amount of pages, we try to select a large number of representative web pages for analysis.

The Alexa top list [1] is a popular starting point for web analyses. The list is assembled using anonymized browsing information collected by browser toolbars, therefore providing a set of representative sites. However, the Alexa top list only provides sites, i.e. domain names. Navigating a browser to these addresses will only yield the primary *landing pages* of the domains; a representative set of addresses will have to include *deep links* as well.

To obtain a set of representative deep links, we collected URLs posted on Twitter using Twitter's stream API, during a two day period in November 2012. To expand shortened URLs, we resolved all URLs through HTTP redirects; we also discarded all URLs that did not resolve to pages of MIME type text/html and HTTP status 200. This left us with a list of 5.4 million unique URLs on 305976 unique domains. We then selected one deep link per domain. To filter against aggressive spammers, we intersect this list of domains with the top 100000 sites from Alexa; this leaves 33747 addresses. From these addresses, we randomly sample 13000 addresses which become half of the addresses we access; the other half is

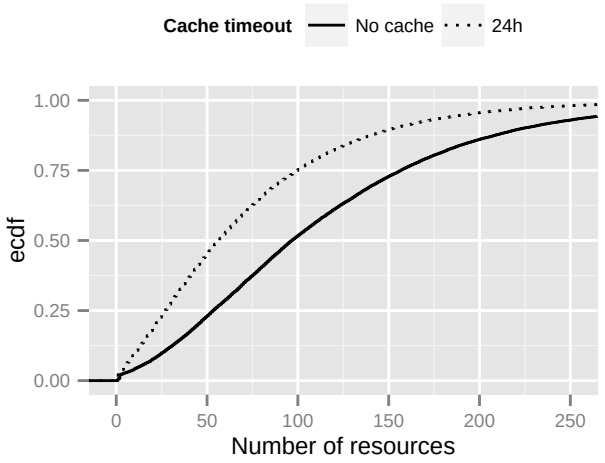

Figure 1: Distribution of number of resources used by web pages.

constructed from the domain names of these deep links. This method of construction allows us to compare the properties of landing pages and deep links.

Browser statistics

To access the web pages, we use each a complete instance of Chromium 22 for Linux. The browser instances are run in a Xvfb X virtual framebuffer server; this provides the browser with the illusion of running on a standard desktop screen output. Every address is accessed with a fresh browser user directory, therefore making the browser start with a clean cache and without cookies.

We control the browser instances using Selenium, and we capture all observed data using Chrome's built-in remote debug interface. This debug interface allows to export many types of operational metrics and information. In particular, we capture events of the Page, Timeline, Console, and Network debugging categories. Before shutting down a browser instance, we also capture the names and types of variables in the global javascript namespace.

In total, one access run produces over 200 GB of uncompressed data metric log files. We then extract and condense information required for this analysis; however, this paper can only scratch the surface of the data collected. The data we collected is freely available on our web page, and we encourage other researchers to make use of it.

3. WEB PAGES IN GENERAL

In this section we analyze web pages for basic properties that are important for browser side processing. All following analysis looks at the entirety of a page, including advertisements. Across the analyzed metrics, we did not find a significant difference between pages on deep links and landing pages. All distributions analyzed present with long tails.

3.1 Requests

Number of requests

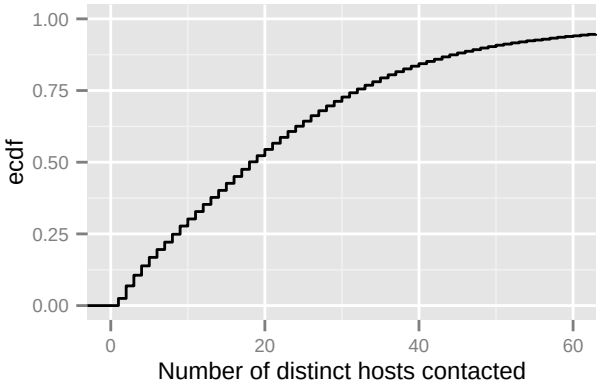

Figure 3: Distribution of number of distinct hosts contacted while loading a web page.

Figure 1 shows the number of HTTP requests executed by the browser: 50 % of web pages load up to 100 resources; the number of resources is roughly uniformly distributed between 0 and 100. The remaining 50 % load more than 100 resources; 15 % even load more than 200 resources. The median amount of redirects per page is 3.8 %, i.e. the number of requests done by the browser is only slightly more than the number of resources loaded.

We analyze the caching headers as provided by the servers and calculate how many requests a browser would do after 24 h, given it would be served the same content, i.e. no dynamic server-side changes would require the browser to load a different set of resources. This value provides a reasonable lower bound on the number of HTTP requests a browser would create. We find that caching has a reasonable effect on the number of requests, halving the number of requests for half of the pages, and reducing the number of requests a bit less for the remaining half of web pages.

To understand the composition of the surprisingly large number of resources per page, we show the distribution of major resource types in Figure 2. The bulk of resources are images, making up about 50 % of the resources on a page, on resource-heavy pages even more than that. The second most frequent resource are javascript (JS) files, at about 25 % of the resources; 50 % of pages contain more than 23 javascript files.

The remaining 25 % of page resources are mostly split between cascading style sheets (CSS) and HTML documents; only 21 % of all pages only contain one HTML document, the remaining pages use iframe and related techniques to combine multiple HTML documents in one page. On the other hand, the top 20 % of pages use more than 15 HTML documents. About 50 % of web pages contain at least one Flash animation, but only 25 % contain more than 4 Flash files.

Number of hosts contacted

We next try to understand how the large number of page requests breaks down on hosts that serve these resources. Figure 3 shows that 50 % of web pages require the browser to contact more than 17 distinct hosts. This result in turn

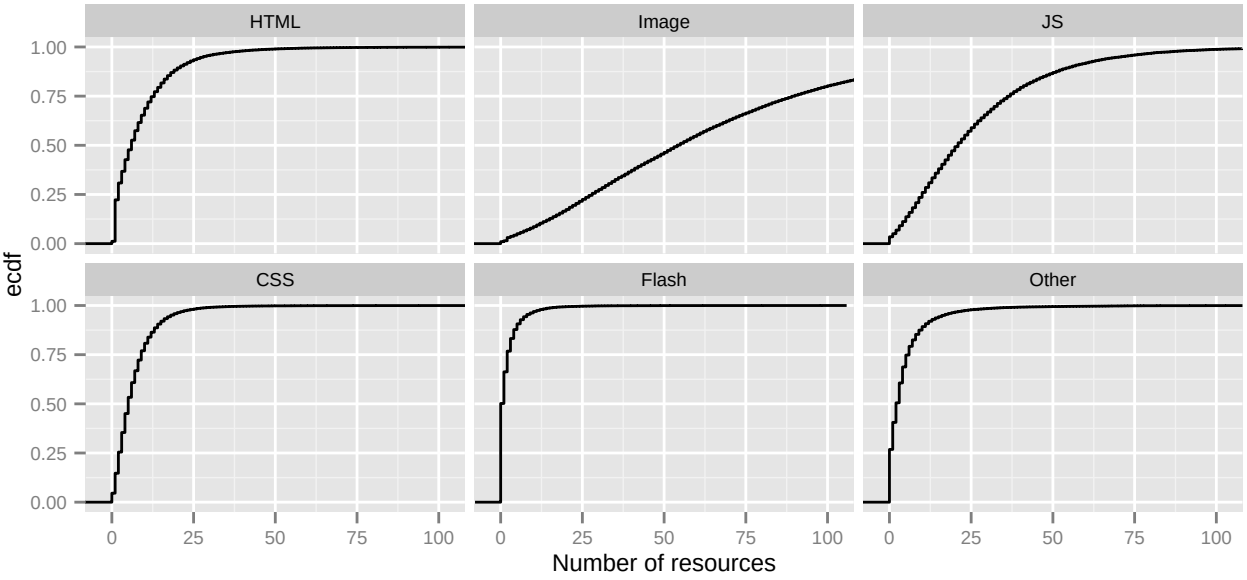

Figure 2: Breakdown of number of resources per web page.

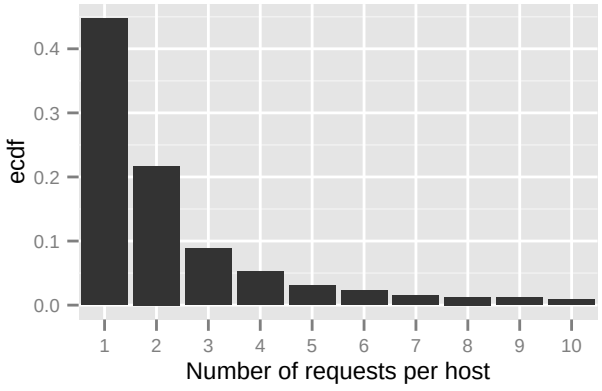

Figure 4: Histogram of the number of requests per host while loading a web page.

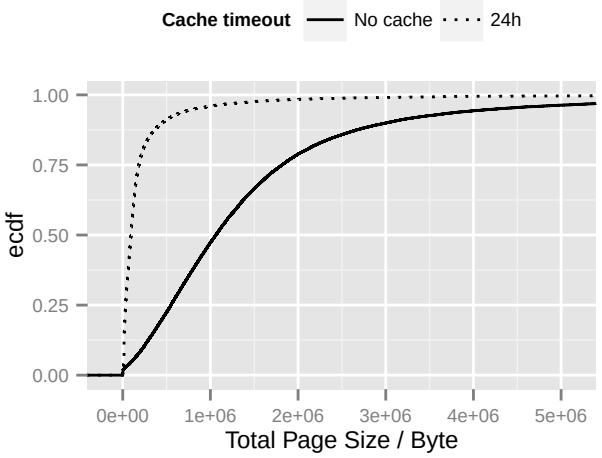

Figure 5: Distribution of total page transfer size.

begs the question how the number of requests is distributed across the number of distinct hosts contacted. Proposals for a new version of HTTP include protocols such as SPDY [3]; many of these proposals for improvement try to address the efficiency of transfers of multiple requests from one server. However, Figure 4 shows that 45 % of all hosts only serve a single requests, and a further 21 % serve two requests.

3.2 Size

We now turn to the second low-level metric, transfer and processing size. HTTP allows clients and servers to negotiate a Content-Encoding that allows the server to compress the requested resource on the fly, commonly using gzip stream compression. We use the term *transfer* size to indicate the amount of data sent over the network, after compression; we use *resource* size to talk about the size of content itself, after decompression by the browser. The *transfer* size is an important metric to judge network utilization, while *resource* sizes are relevant for browser side processing load.

Page size

Figure 5 shows the distribution of web page transfer sizes: 75 % of web pages are between 0 and 1.8 MB, with sizes roughly uniformly distributed; 75 % are larger than 500 kB, and 50 % of pages are larger than 1 MB.

Caching, however, has an enormous effect on the transfer size of a web page, reducing it by over an order of magnitude for accesses after 24 h; accesses in a shorter time frame save even more. This effect is even compounded when accessing multiple pages from the same web site.

The amount of data the browser finally has to process is about twice the transfer size; we show the breakdown of aggregate resource sizes in Figure 6. Even though images make up 50 % of resources, they only make up 26 % of the resource data processed (median 450 kB); this is because the

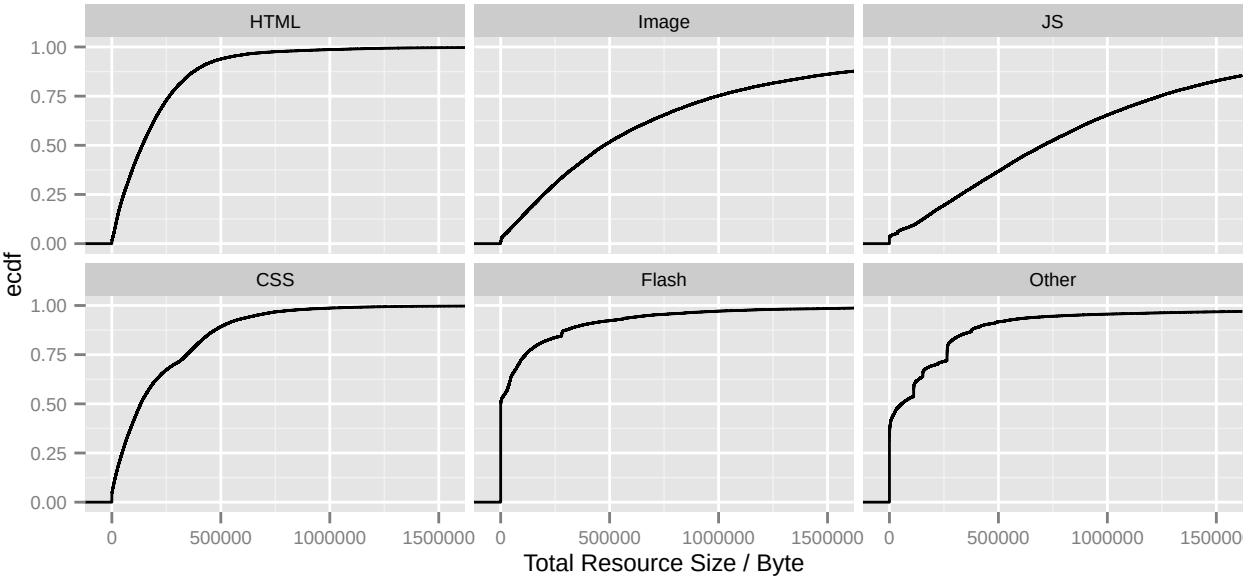

Figure 6: Breakdown of resource sizes per page.

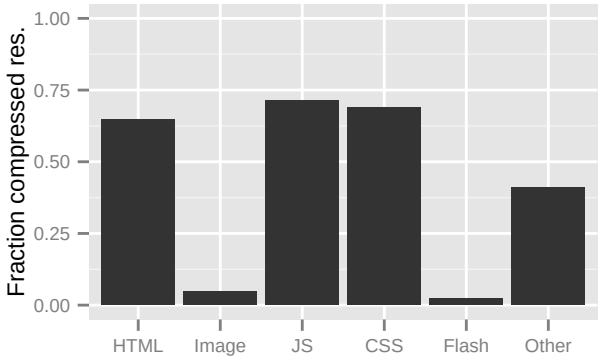

Figure 7: Fraction of resources being served with server-side compression enabled (Content-Encoding).

median size of an image is only 2.3 kB (not shown). The largest portion of resource size is presented by javescript files, which make up for 39 % of the size processed (median 730 kB); the median size of a javascript resource is 9 kB.

Both HTML and CSS resources only make up for each 7 % of the total resource size (each median 140 kB); however, the median file size for HTML files is 12.5 kB, while the median CSS size is only 6.3 kB.

Compression

The large discrepancy between transfer size and resource size already indicates a widespread use of compression; Figure 7 shows how frequently a Content-Encoding is being used. For the important text-based file formats, about 73 % of the files were transferred with compression, with over 99.8 % using gzip encoding, and the rest using deflate or invalid encodings, such as UTF-8. 83 % of hosts that compress one file of a given MIME type compressed all files of this type; this indicates fixed encoding rules in their configuration. However, for one quarter of all compressed files, the absolute compression gains are less than 740 B, less than a full-size MTU Ethernet packet. Because compression and decompression creates processing overhead on both server and browser side, and active Content-Encoding can interact with web caches, we suggest a policy that will weigh the benefits of space reduction with the associated processing overhead.

Peculiar Tracking

When analyzing the URL length of images, GIF images exhibit a behavior that is not mirrored in either JPEG or PNG, the two other popular image formats: a significant fraction of GIF images have a very long URL length. Figure 8 shows the results of further investigation: not only do many GIF images come from long URLs, but these files at the same time are exceptionally small. About 20 % of all GIF images are both smaller than 50 B and have URLs longer than 150 characters; other image formats do not exhibit this behavior.

Manual inspection of sample URLs shows that this peculiarity can be attributed to user tracking methods: the long URLs contain leaked browsing information in a query string, and the returned image data is most likely a single transparent pixel.

4. ADVERTISEMENTS

We now turn to analyze the prevalence and impact of advertisements. We first describe our methodology of classifying resources as advertisements, and then present statistical results on advertisements on web pages.

4.1 Classification of Ads

We use the Adblock Plus extension to distinguish between "regular" page resources and resources related to ads. As classification database, we use "Fanboy's List" in Adblock Plus; this results in a slightly conservative classification, because this blocking database focuses on the English speaking web. Furthermore, other 3rd-party web elements such as

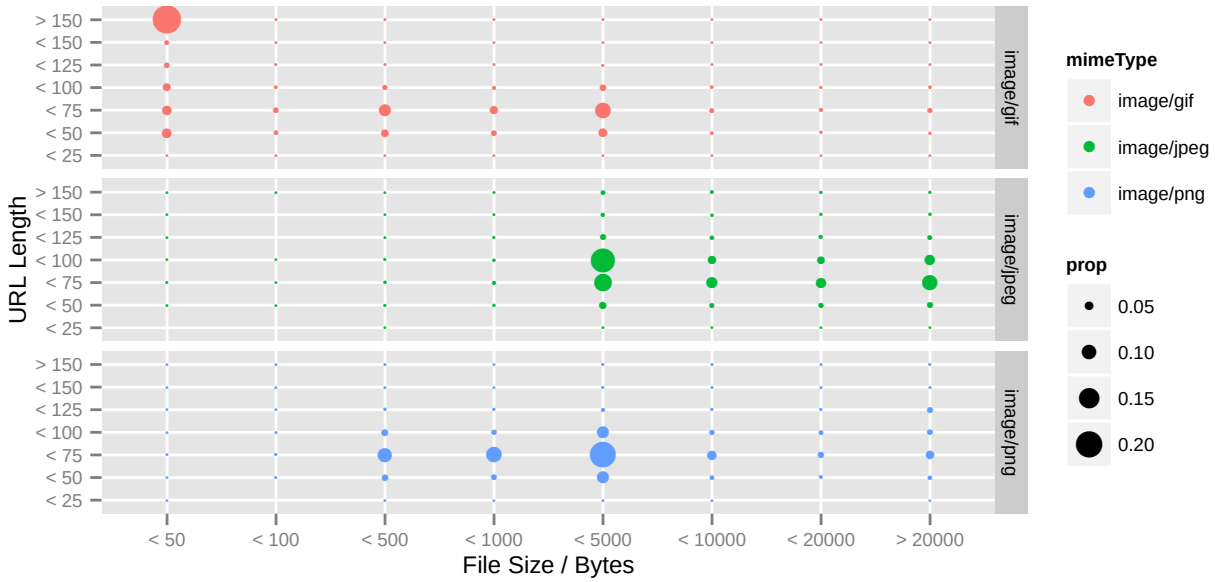

Figure 8: Distribution of file size and URL length for popular image formats. Other than JPEG and PNG, many GIF images have both extremely small file size and long URLs at the same time.

Class	Description
common	requests that exist both in A and B and were not blocked in B.
ads	requests that exist both in A and B and were blocked in B.
soft-common	requests that do not match exactly between A and B, but are similar; the request in B was not blocked.
soft-ads	requests that do not match exactly between A and B, but are similar; the request in B was blocked.
unknown-ads	only exists for B; requests that were blocked, but have no partner in A.
unknown	requests that exist only in A or B but have no similar partner. These requests are certainly no ads in B, but are either ads or no ads in A.

Table 1: Classification of resources based on a capture with a standard browser (A = ads) and a capture with Adblock (B = block).

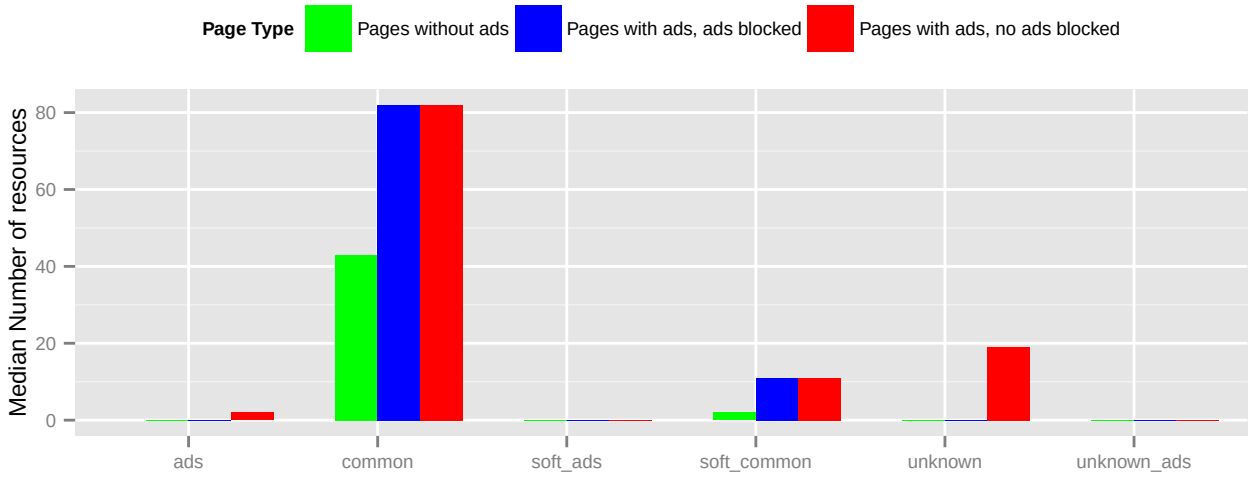

Figure 9: Breakdown of ad resource classifications (see Table 1). Most URLs match verbatim between *adblock* and *vanilla* runs (*common*, *ads*), and a few require string similarity matching (*soft_common*, *soft_ads*). On ad-free pages, all URLs can be matched verbatim or using string similarity, leaving no *unknown* URLs.

user tracking, analytics, or social media integration are also not classified as ads.

We modified the Adblock Plus extension slightly to explicitly report all URLs that are being blocked. We capture this information and merge it with the captured browser metrics to mark resources as advertisements. Unfortunately this approach is not sufficient to capture all resources and classify them; this is because the loading of ads is often a multi-stage process that is driven by javascript code included by the main HTML page. Because Adblock will block requests for ads right away, and because ads can be loaded in multiple stages, we can not observe all requests that would occur in a session running without Adblock. We therefore access every web page once with Adblock enabled, and once without Adblock (*vanilla*); this yields two sets of resource addresses: a full *vanilla* set, and a smaller *adblock* set that has some requests tagged as being ads.

We then combine the information of both *adblock* and *vanilla* runs to determine whether a request in the full *vanilla* set is an ad or not. We assign every request to one class, as described in Table 1: if a given URL was requested in both *adblock* and *vanilla* runs, it is assigned to the *common* class if it was not blocked in *adblock*, or to the *ads* class if it was blocked. Because repeat accesses to the same web page often generate slightly different resource URLs (e.g. because of tracking scripts), some URLs in *vanilla* and *adblock* require a fuzzy string match; these requests are assigned the classes *soft-common* or *soft-ads*, depending on whether they were blocked in *adblock*.

Finally, all remaining requests are assigned to the *unknown* or *unknown-ads* classes. These are the requests that did only occur once in either *adblock* or *vanilla*, but not in both. Figure 9 shows the breakdown of the number of requests per domain. The breakdown shows that for ad-free pages (green bars), and in pages with ads (blue line) there are no requests in the *adblock* run that are classified as *unknown*. On the other hand, *vanilla* runs on pages with ads contain a large number of *unknown* resources. This indicates that that most of these resources in fact belong to ads, but were not observed during the *adblock* session due to other resources being blocked. This is a fair assumption, because ad-free pages and *adblock* runs only exhibit a very small number of *unknown* resources: in *adblock*, less than 25 % of pages contain one or more *unknown* resources. We therefore treat *unknown* resources as advertisements.

4.2 Ad Statistics

We find that more than 71 % of the web pages we analyzed contain ads. On these pages, the ads make up for 20 % of all HTTP requests, and for 12.5 % of all data transferred.

Figure 10 and Figure 11 show the distribution of number of resources and the total size of ads for web pages that contain ads. The median number of ad resources per web page is 23, and their median aggregate transfer size sums up to 137 kB; 25 % of web pages with ads load more than 40 resources at over 280 kB aggregate transfer size (580 kB aggregate resource size).

Ad-Free Pages vs. Pages Without Ads

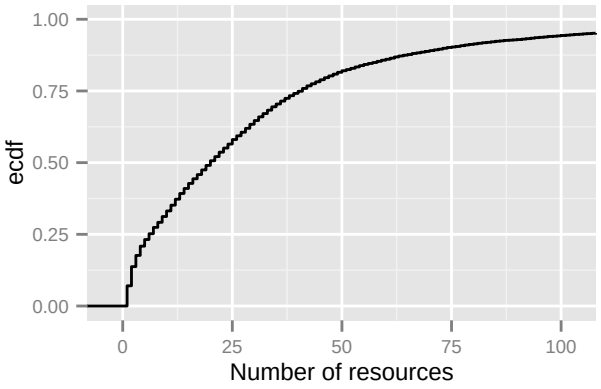

Figure 10: Distribution of number of ad-related resources on pages with ads.

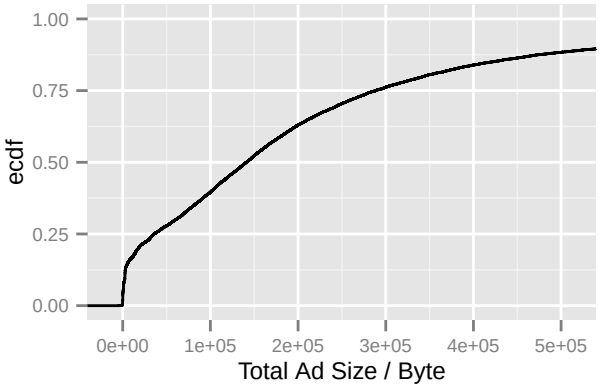

Figure 11: Distribution of total transfer size of ads per domain.

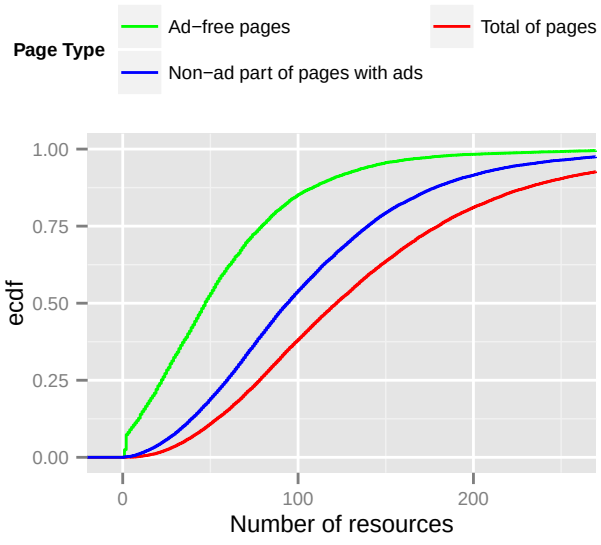

Figure 12: Distribution of number of resources per domain.

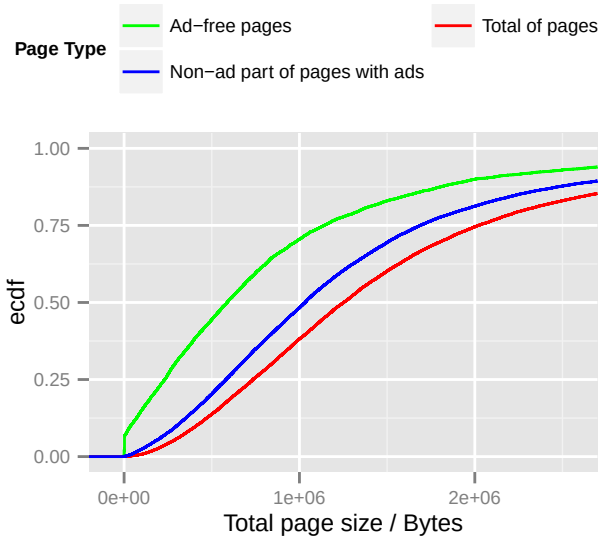

Figure 13: Distribution of total transfer size per domain.

Surprisingly, ad-free web pages, and web pages with ads blocked exhibit significantly different distributions for the basic metrics of size and number of resources. Figure 12 and Figure 13 compare these metrics for ad-free pages (green line), pages with ads (red line), and pages with ads, but their ads removed (blue line). In both metrics, pages without ads contain significantly fewer resources, and are also significantly smaller: the median ad-free page contains half as many resources (47) as a page with ads blocked (94) and is 43 % smaller than its counterpart (587 kB vs 1 MB).

We believe that this significant difference might be related to the incentive model of web advertisements: pages with ads try to increase the exposure of users to these ads; one way to do so is increase the stay of a user on the web site. By providing more content previews and in total a richer experience, these web sites might try to engage the user more. Sites that are ad-free by choice are not exposed to these incentives, which leads to leaner pages.

4.3 Background activity of ads

We finally look at the number of javascript timer events while loading and displaying web pages. This is an important metric, because it relates directly to CPU wakes and therefore to the consumption of energy of the system. Modern processors can sleep in extremely low power modes; however, deeper and lower power modes require increasingly long sleep periods. Transition from and to sleep modes also requires a significant amount of energy.

Figure 14 shows the distribution of the number of javascript timer events during a 40 s stay per web page. Other than in the case of number of resources and size, ad-free pages and pages with ads blocked exhibit almost the same behavior; the remaining difference might be because of our conservative approach of blocking ads. However, pages with ads show a completely different behavior: while the median page without ads produces 152 events during the 40 s stay, the median page with ads produces more than 600 events; this corresponds to an average inter event time of 263 ms vs 67 ms. Across almost the whole distribution, corresponding quantiles of pages with ads exhibit more than 3 times as many events as pages with ads blocked, i.e. browsing with ads blocked significantly reduces the number of timer events to 1/3.

We analyze only javascript timer events, as set by setInterval() and setTimeout(); we did not capture wake-ups due to animated GIF images, video playback, or flash animations. We expect that an analysis of these causes for wake-ups would draw a similar picture.

5. POPULAR DOMAINS

We now look at the prevalence of 3rd party services on web pages. For each accessed page, we collect a list of domains, based on the URLs of the accessed resources. We then determine the prevalence of each domain across all accessed web pages. Figure 15 shows the 28 most prevalent domains, sorted by their prevalence. Most important, *Google Analytics* can be found on over 80 % of web pages; the domains of various Google services also make up over 1/3 of the top 28 domains shown. Many web pages use social media integration, making Facebook and Twitter take second and third place in prevalence.

6. CONCLUSION

We have presented an initial statistical analysis of a large set of web pages. Accessing these pages by an actual browser allowed us to observe properties of the pages that might otherwise be missed, such as dynamic requests generated by javascript. Our analysis shows that metrics on the web always come in a long tail distribution, and web pages in general are fairly large, thereby incurring significant processing on the browser side.

We also find that advertisements on the web are extremely common, and that ads also seem to incentivize the creation of larger web pages. Moreover, ads themselves make up a sizable fraction of web pages. Lastly, ads incur a massive overhead in background processing. Given that around 9 % of web users use some form of ad blocking [2], it is important that both web page owners and advertisement providers reduce the resource consumption of ads to avoid alienating their users.

All resources and data we collected are available on our web page, `http://labos.epfl.ch/webstats`. We encourage other researchers to further explore this data and validate our analysis.

7. REFERENCES

[1] Alexa Top 1,000,000 Sites. `http://www.alexa.com/topsites`.

[2] Clicked off: Doom beckons for online ads, The Economist. `http://www.economist.com/news/international/21565931-doom-beckons-online-ads-clicked`.

[3] SPDY (Chromium Developer Documentation). `http://dev.chromium.org/spdy`.

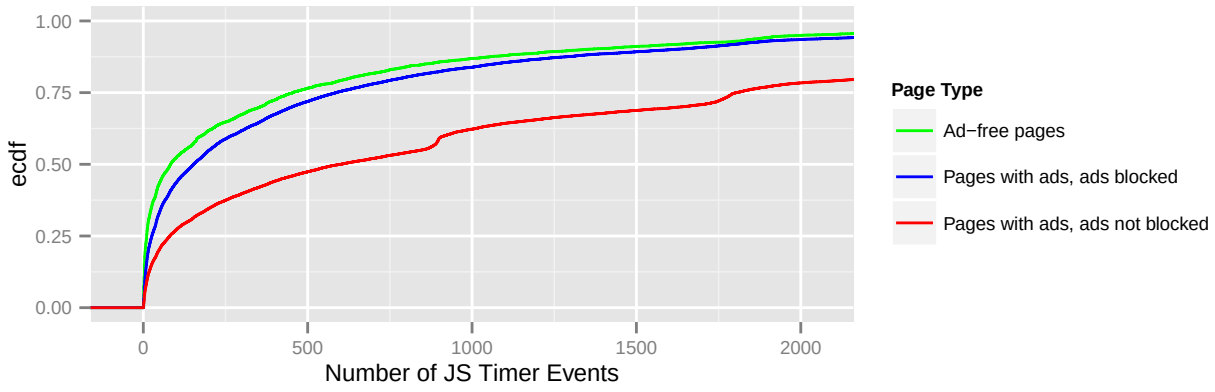

Figure 14: Distribution of number of javascript timer events delivered during a 40 second stay on web pages.

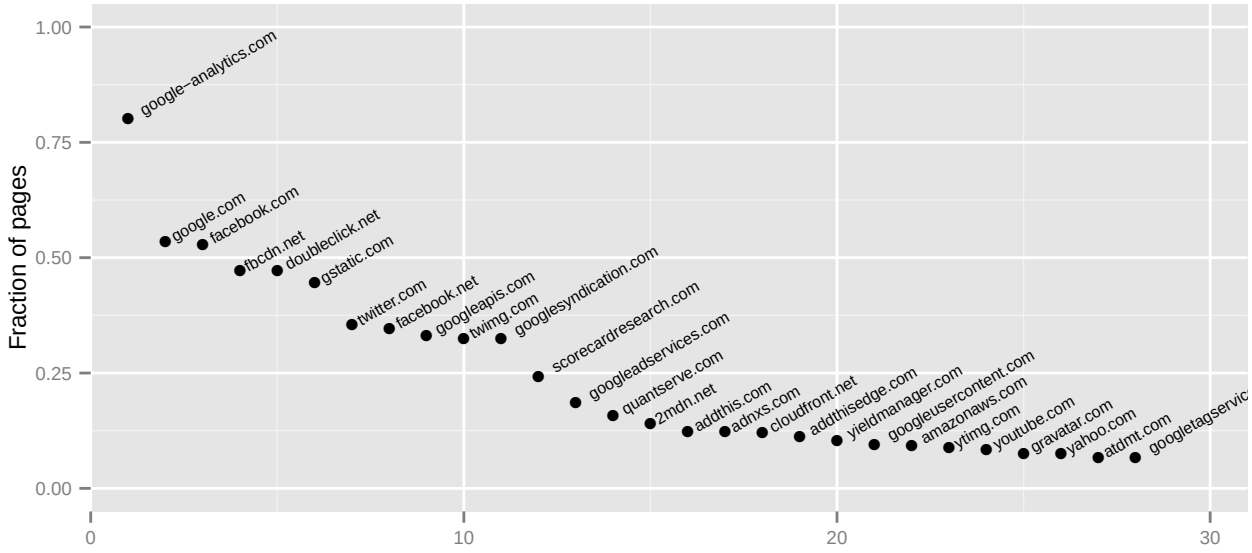

Figure 15: List of domains most frequently referenced by web pages. The vertical axis denotes what fraction of web pages a given domain is used by; e.g. google-analytics.com is accessed by 80.2% of the pages analyzed.